Triedenie

- proces preusporiadania danej množiny objektov v špecifickom *poradí*. Účelom triedenia je uľahčiť neskoršie vyhľadávanie prvkov triedenej množiny.

Triedenie polí

Od metód triedenia polí by sme mali predovšetkým požadovať úsporné využívanie pamäti, ktorú máme k dispozícii. To znamená, že permutácia, ktorou sa prvky usporadúvajú, sa má vykonať na mieste (t.j. bez pomocnej pamäte) a metódy, ktoré prenášajú prvky z poľa a do výsledného poľa b, sú svojou povahou menej zaujímavé.

Teraz prejdeme k prvej klasifikácii podľa efektívnosti metód, t.j. podľa ich časovej náročnosti.

Dobré *meradlo efektívnosti* dostaneme spočítaním počtov C potrebných I*porovnaní hodnôt* M *presunov prvkov*. Tieto počty sú funkciami počtu n prvkov, ktoré sa majú triediť. Pretože dobré triediace algoritmy vyžadujú rádovo n.log n porovnaní, rozoberieme najprv niekoľko jednoduchých a zrejmých triediacich metód nazývaných *priame metódy*, z ktorých všetky vyžadujú rádovo n^2 porovnaní kľúčov.

Metódy triedenia, ktoré triedia prvky na mieste, možno rozdeliť na tri základné kategórie podľa toho, z ktorej metódy vychádzajú:

1. *Triedenie vkladaním.*

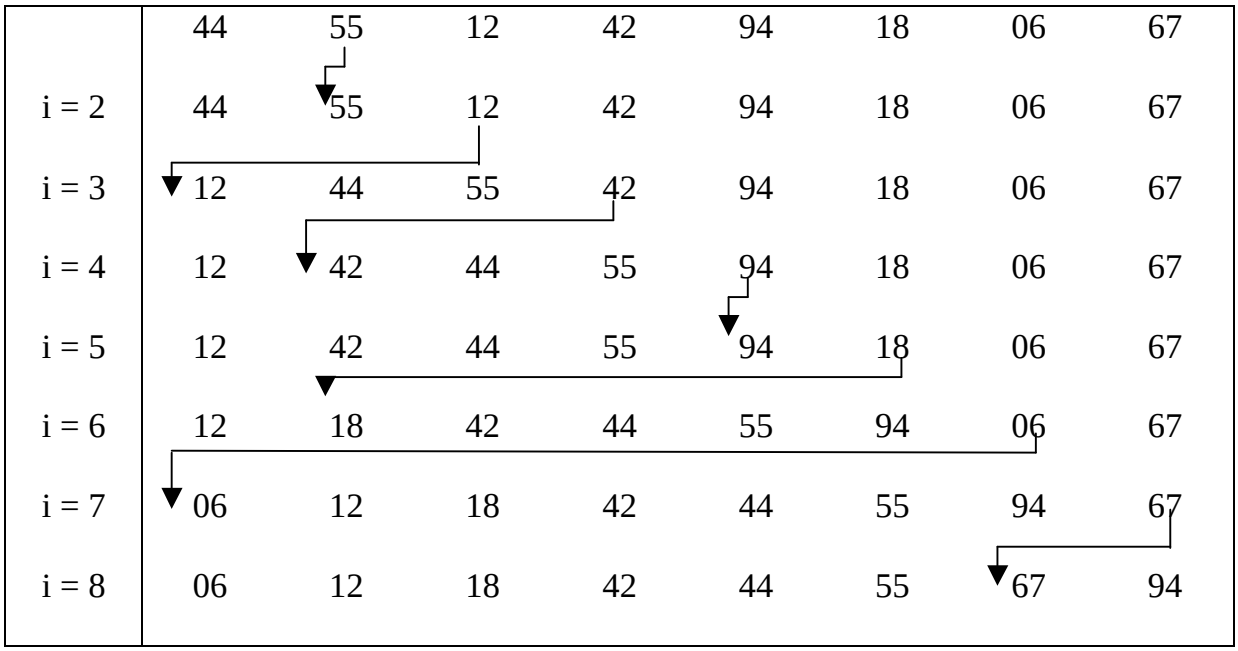
2. *Triedenie výberom.*

3. *Triedenie výmenou.*

Triedenie priamym vkladaním

Túto metódu používajú hráči kariet. Prvky (v tomto prípade karty) sa rozdelia na dve skupiny - postupnosti: zdrojovú a cieľovú. Z prvej - zdrojovej - sa prvky vyberajú, do druhej - cieľovej - sa ukladajú. Zdrojovú postupnosť tvoria prvky $a_i,...,a_n$, cieľovú prvky $a_1,..., a_{i-1}$. V rámci každého kroku algoritmu sa zo zdrojovej postupnosti vyberie i-tý prvok a vloží sa na patričné miesto cieľovej postupnosti. Algoritmus sa začína hodnotou indexu i = 2 a v rámci každého kroku sa jeho hodnota zväčší o jednotku.

Príklad triedenia:



Algoritmus triedenia priamym vkladaním má takýto tvar:

```
for i:=2 to n do  begin
                      x:=a[i];
                      "vlož x na patričné miesto
                      v postupnosti a1 . . ai"
                  end
```

Pri hľadaní vhodného miesta v cieľovej postupnosti je účelné striedať porovnania prvkov s ich presunmi. Prvok *x* sa dostane na svoje miesto v cieľovej postupnosti tak, že ho porovnáme s jeho ľavým susedom *a*. Ak je väčší, tak tohto suseda posunieme o jedno miesto vpravo.

Proces hľadania vhodného miesta pre prvok *x* končí vtedy, ak:

1. Hodnota prvku a_j je menšia ako hodnota prvku *x*.
2. Dosiahneme ľavý koniec cieľovej postupnosti.

```
procedure triedenievkladaním;
  var i,j:index;x:prvok;
begin
  for i:=2 to n do
     begin
       x:=a[i]; j:=i-1;
       while x<a[j] do
          begin
            a[j+1]:=a[j];
            j:=j-1;
          end;
       a[j+1]:=x;
     end;
end;
```

Analýza triedenia priamym vkladaním

Počet porovnaní hodnôt C_i v i-tom prechode je najviac i-1 a najmenej 1. Za predpokladu, že všetky permutácie *n* hodnôt sú rovnako pravdepodobné, môžeme C_i v priemere pokladať za rovnajúce sa *i/2*. Počet presunov M_i (priradením jednotlivým prvkov) je $C_i + 2$ vrátane zarážky. Celkový počet porovnaní a presunov potom bude

$C_{min} = n - 1$ $\qquad$ $M_{min} = 2.(n - 1)$

$C_{priem} = \frac{1}{4}\left(n^2 + n - 2\right)$ $\qquad$ $M_{priem} = \frac{1}{4}\left(n^2 + 9n - 10\right)$

$C_{max} = \frac{1}{2}\left(n^2 + n\right) - 1$ $\qquad$ $M_{max} = \frac{1}{2}\left(n^2 + 3n - 4\right)$

Najmenšie hodnoty C a M sú v prípade, ak zdrojová postupnosť obsahuje usporiadané prvky. Tento prípad nazývame najlepší. Opakom je najhorší prípad, ktorý nastane vtedy, keď sú prvky zdrojovej postupnosti v obrátenom poradí (opačne usporiadané). V tomto zmysle triedenie vkladaním dokumentuje skutočne reálnu situáciu.

Keď si uvedomíme, že cieľová postupnosť $a_1 \ldots a_{i-1}$, t.j. tá, do ktorej sa vybratý prvok ukladá, je utriedená, tak je možné algoritmus triedenia vkladaním veľmi jednoducho vylepšiť. Zlepšenie sa týka rýchlejšej metódy určenia miesta v cieľovej postupnosti, do ktorého sa má prvok uložiť. Obyčajne sa volí metóda binárneho vyhľadávania, ktorá hľadané miesto nájde postupným delením postupnosti na dve podpostupnosti. Modifikovaný algoritmus sa nazýva *binárne vkladanie*:

```
procedure binárnetriedenie;
  var i,j,l,r,m:index; x:prvok;
begin
  for i:=2 to n do
     begin
       x:=a[i];l:=1;r:=i-1;
       while l≤r do
          begin
            m:=(l+r) div 2;
            if x<a[m] then r:=m - 1
                      else l:=m + 1;
          end;
       for j:=i-1 downto l do a[j+1]:=a[j];
       a[l]:=x;
     end;
end;
```

Analýza binárneho vkladanie

Miesto pre uloženie prvku sa nájde vtedy, keď platí: $a[j] \le x < a[j+1]$.

Teda skúmaný interval sa nakoniec rovná 1; čiže interval pozostávajúci z *i* hodnôt sa rozpoľuje *t* krát, kde *t* je horná časť $\log_2 i$. Počet porovnaní potom bude $C = \sum_{i=1}^{n} t$.

Aproximáciou tejto sumy pomocou integrálu dostávame:

$\int_1^n \log x \, dx = \left[x(\log x - c)\right]_1^n = n(\log n - c) + c$, pričom c = log e = 1/ln 2= 1,44269...

Počet porovnaní je v podstate nezávislý od začiatočného usporiadania prvkov. Ale vzhľadom na skracovací charakter delenia zahrnutého v rozpoľovaní skúmaného intervalu môže byť skutočný počet porovnaní potrebných pri *i* hodnotách až o 1 vyšší, ako sa očakávalo. Podstata tejto systematickej odchýlky spočíva v tom, že miesto na uloženie prvku sa v priemere rýchlejšie zistí na ľavom konci postupnosti ako na pravom konci. Preto sa uprednostňujú tie prípady, kedy sú začiatočné postupnosti značne neusporiadané. Najmenší počet porovnaní je potrebný vtedy, keď sú prvky v zdrojovej postupnosti opačne usporiadané, najviac porovnaní treba pri usporiadanej zdrojovej postupnosti. Tieto črty charakterizujú neprirodzené správanie algoritmu triedenia.

Žiaľ, vylepšenie algoritmu triedenia binárnym vyhľadávaním sa týka iba počtu porovnaní, a nie počtu potrebných presunov. Presuny prvkov vyžadujú vo

všeobecnosti oveľa viac času ako porovnanie dvoch prvkov. Uvedeným vylepšením algoritmu sa výrazne nevylepší hodnota ukazovateľa M: tento naďalej zostáva rádu n_2.

V skutočnosti triedenie už utriedeného poľa zaberá viac času ako metóda priameho vkladania prostredníctvom sekvenčného vyhľadávania! Napokon triedenie vkladaním nepatrí medzi najvýhodnejšie metódy; vloženie prvku s nasledujúcim posunom všetkých prvkov o jednu pozíciu je neekonomické. Lepšie výsledky by sme očakávali od takej metódy, pri ktorej sa jednotlivé prvky iba presúvajú na dlhšie vzdialenosti.

Táto úvaha vedie k druhej metóde triedenia, ktorá sa nazýva triedenie metódou výberu.

Triedenie priamym výberom

Táto metóda je založená na nasledujúcom princípe:

1. Vyber prvok s najmenšou hodnotou.
2. Vymeň ho s prvým prvkom a_1.
3. Potom tieto operácie opakuj s ostávajúcimi n - 1 prvkami, s n - 2 prvkami atď., až zostane jediný najväčší prvok.

44	55	12	42	94	18	**06**	67
06	**55**	**12**	42	94	18	44	67
06	12	**55**	42	94	**18**	44	67
06	12	18	**42**	94	55	44	67
06	12	18	42	**94**	55	**44**	67
06	12	18	42	44	**55**	94	67
06	12	18	42	44	55	**94**	**67**
06	12	18	42	44	55	67	94

Program môžeme zapísať takto:

```
for i:=1 to n-1 do
  begin
     "priraď index najmenšiemu prvku postupnosti a[i]..a[n] do k;"
  end
```

Táto metóda sa nazýva priamy výber a je v istom zmysle opakom metódy priameho vkladania. Pri priamom vkladaní sa v rámci každého kroku uvažuje jedna položka

zdrojovej postupnosti a všetky prvky cieľovej postupnosti na vyhľadanie miesta vloženia. Pri priamom výbere sa v každom kroku berie do úvahy celá zdrojová postupnosť, z ktorej sa vyberie prvok s najmenšou hodnotou a uloží sa ako ďalší prvok do cieľovej postupnosti.

```
procedure triedenievyberom;
  var i,j,k:index;x:prvok;
begin
  for i:=1 to n-1 do
     begin
          k:=i;x:=a[i];
          for j:=i+1 to n do
               if a[j]<x then
                    begin
                         k:=j;x:=a[j];
                    end;
          a[k]:=a[i];a[i]:=x;
     end;
end;
```

Analýza triedenia priamym výberom

Je zrejmé, že počet porovnaní hodnôt C nezávisí od začiatočného usporiadania hodnôt. V tomto zmysle možno túto metódu považovať za menej prirodzenú ako triedenie vkladaním. Počet porovnaní $C=\frac{1}{2}\left(n^2-n\right)$. Počet presunov M je minimálne $M_{min}=3(n-1)$ ak sú hodnoty v zdrojovej postupnosti usporiadané, a maximálne $M_{max}=\text{trunc}\left(\frac{n^2}{4}\right)+3(n-1)$ ak sú hodnoty v zdrojovej postupnosti usporiadané opačne.

Záverom môžeme konštatovať, že vo všeobecnosti je triedenie priamym výberom efektívnejšie ako triedenie priamym vkladaním, hoci v prípadoch, keď sú hodnoty v zdrojovej postupnosti usporiadané alebo takmer usporiadané, je triedenie priamym vkladaním predsa len o niečo rýchlejšie.

Triedenie priamou výmenou

Nasledujúci algoritmus triedenia výmenou je založený na princípe postupného porovnávania a zámeny dvojíc susedných prvkov až dovtedy, kým všetky prvky postupnosti nie sú utriedené. Podobne ako v predchádzajúcom prípade triedenia aj táto metóda spočíva v cyklickom spracúvaní poľa. V rámci každého prechodu sa vyberie najmenší prvok skúmanej podmnožiny, ktorý sa presunie na ľavý koniec poľa. Ak by sme si sledované pole predstavili namiesto v horizontálnej polohe vo vertikálnej polohe a, s tou istou dávkou predstavivotsti, triedené prvky ako bubliny vo vodnej nádrži (so špecifickou váhou úmernou hodnote), tak v rámci každého prechodu poľom dôjde k "vybublaniu" vybratého prvku na úroveň zodpovedajúcu jeho váhe.

	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7	i = 8
44	06	06	06	06	06	06	06
55	44	12	12	12	12	12	12
12	55	44	18	18	18	18	18
42	12	55	44	42	42	42	42
94	42	18	55	44	44	44	44
18	94	42	42	55	55	55	55
06	18	94	67	67	67	67	67
67	67	67	94	94	94	94	94

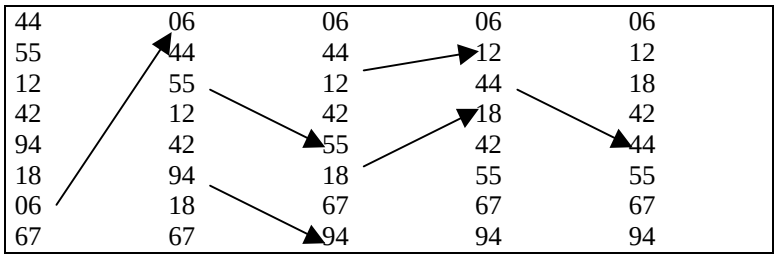
Táto metóda je známa pod pojmom *bublinové triedenie*.

```
procedure bublinovetriedenie;
  var i,j:index;x:polozka;
begin for i:=2 to n do
          begin for j:=n downto i do
                    if a[j-1]>a[j] then
                                  begin
                                       x:=a[j-1];
                                       a[j-1]:=a[j];
                                       a[j]:=x;
                                  end;
          end;
end;
```

Je zrejmé, že tento algoritmus sa dá zlepšiť. Keď si všimneme príklad uvedený vyššie, uvidíme, že v posledných troch prechodoch sa prvky nepremiestňovali, pretože už boli utriedené. Ak si v rámci prechodov dokážeme zapamätať, či nastala alebo nenastala nejaká výmena prvkov, je možné realizovať prvé zlepšenie algoritmu, ktoré spočíva v presnom stanovení konca triediaceho procesu. Algoritmus končí v tom okamihu, keď v rámci posledného prechodu nenastala operácia výmeny prvkov. Toto zlepšenie sa dá ďalej vylepšiť jednoducho tým, že okrem registrovania poslednej výmeny si zapamätáme aj polohu (index) poslednej výmeny. Napríklad je jasné, že všetky dvojice susedných prvkov pod takýmto indexom *k* sú v žiadanom poradí. Nasledujúci prechod môže preto končiť pri tomto indexe namiesto toho, aby pokračoval až po vopred určený dolný index *i*.

Šikovný programátor si iste všimne určitú asymetriu: nesprávne umiestnená bublinka na "ťažkom" konci ináč utriedeného poľa sa dostane do svojej polohy jediným prechodom, kým neumiestnená bublinka na opačnom konci bude putovať k svojej správnej pozícii v každom prechode iba o jeden krok.

Napríklad pole 12 18 42 44 55 67 94 06 sa utriedi vylepšeným bublinovým triedením jediným prechodom, kým pole 94 06 12 18 42 44 55 67 bude vyžadovať až sedem prechodov. Táto neprirodzená asymetria nás nabáda k tretiemu vylepšeniu: striedať smer prehľadávania v jednotlivých (po sebe idúcich) prechodoch. Takáto stratégia sa nazýva *triedenie pretriasaním (shakersort)*.




```
procedure shakesort;
     var j,k,l,r:index;x:prvok;
begin l:=2;r:=n;k:=n;
     repeat
      for j:=r downto l do
          if a[j-1]>a[j] then
               begin
                    x:=a[j-1];a[j-1]:=a[j];a[j]:=x;
                    k:=j;
               end;
      l:=k+1;
      for j:=l to r do
          if a[j-1]>a[j] then
               begin
                    x:=a[j-1];a[j-1]:=a[j];a[j]:=x;
                    k:=j;
               end;
      r:=k-1;
     until l>r;
end;
```

Analýza bublinového triedenia a triedenia pretriasaním

Počet porovnaní hodnôt pri triedení priamou výmenou je $C=\frac{1}{2}\left(n^2-n\right)$ a minimálne, priemerné a maximálne počty presunov (priradení medzi prvkami) sú $M_{\min}=0, M_{priem}=\frac{3}{4}\left(n^2-n\right), M_{\max}=\frac{3}{2}\left(n^2-n\right).$

Minimálny počet porovnaní je C_{min} = n - 1. Poznamenávame však, že všetky uvedené zlepšenia nijako neovplyvňujú počet výmen, ale len zmenšujú počet

nadbytočných dvojnásobných kontrol. Zámena dvoch prvkov je však väčšinou oveľa nákladnejšou operáciou než porovnanie hodnôt; naše vynikajúce zlepšenia majú preto oveľa menší účinok, než by človek intuitívne očakával.

Táto analýza dokazuje, že triedenie výmenou a jeho malé zlepšenia nedosahujú kvalitu triedenia priamym vkladaním ani triedenia priamym výberom a bublinové triedenie môže v skutočnosti sotva čo ponúknuť okrem príťažlivého názvu. Algoritmus triedenia pretriasaním sa používa s výhodou v tých prípadoch, v ktorých je známe, že prvky sú už takmer usporiadané. Je to však v praxi zriedkavý prípad.

Porovnanie metód vnútorného triedenia

Symbolom *n* označujeme počet triedených prvkov, symbol *C* a *M* vyjadrujú celkové počty potrebných porovnaní a presunov prvkov. Pomocou týchto symbolov vyjadríme analytické vzorce pre všetky tri priame triediace metódy.

	min	priem	max
Priame vkladanie	C = n - 1 M=2(n - 1)	$(n^2+n-2)/4$ $(n^2-9n+10)/4$	$(n^2-n)/2-1$ $(n^2+3n-4)/2$
Priamy výber	C = $(n^2-n)/2$ M= $3(n-1)$	$(n^2-n)/2$ $n(\ln n+0{,}57)$	$(n^2-n)/2$ $n^2/4+3(n-1)$
Priama zmena (bublinové triedenie)	C= $(n^2-n)/2$ M=0	$(n^2-n)/2$ $(n^2-n)*0{,}75$	$(n^2-n)/2$ $(n^2-n)*1{,}5$

Uvedené vzorce poskytujú iba približnú mieru účinnosti algoritmov triedenia. Udávajú efektívnosť ako funkciu celkového počtu prvkov *n* a umožňujú delenie triediacich algoritmov na jednoduché priame metódy (n^2) a prepracované alebo logaritmické metódy (n . log n). Pri analýze algoritmov sme nebrali do úvahy zložitosti iných operácií ako napr. porovnanie hodnôt a presuny prvkov a riadenie cyklu.

V nasledujúcej tabuľke sú uvedené časy výpočtov jednotlivých algoritmov triedenia (v milisekundách), implementovaných v jazyku pascal na počítači CDC6400. Nachádzajú sa v nej časové údaje pre triedenie už utriedeného poľa, náhodne usporiadaného a obrátene usporiadaného poľa.

	Utriedené		Náhodne usporiadané		Obrátene usporiadané	
Priame vkladanie	12	23	366	1444	704	2836
Binárne vkladanie	56	125	373	1327	662	2490
Priamy výber	489	1907	509	1956	695	2675
Bublinové triedenie	540	2165	1026	4054	1492	5931
Bublinové triedenie *	5	8	1104	4270	1645	6542
Triedenie pretriasaním	5	9	961	3642	1619	6520
Shellovo triedenie	58	116	127	349	157	492

* bublinové triedenie s indikátorom - indikátor registruje poslednú výmenu prvkov v poli. Jeho zavedením sa zvýšila celková účinnosť bublinového triedenia.

Každý stĺpec v tabuľke sa navyše skladá z dvoch podsĺpcov, z ktorých ľavý udáva časy pre 256 triedených prvkov, pravý pre 512 prvkov.

Pre záujemcov o ďalšie formy triedení spomeniem napr. Quicksort. Ide o logaritmickú metódu (n . log n). S radosťou im poskytnem študijný materiál